Scheduling Algorithms for a Cache Pre-Filling Content Distribution Network

Reuven Cohen Liran Katzir Danny Raz
Department of Computer Science
Technion
Haifa 32000, Israel

Abstract— **Cache pre-filling is emerging as a new concept for increasing the availability of popular web items in cache servers. According to this concept, web items are sent by a "push-server" to the proxy cache servers, usually through a broadcast-based or a multicast-based distribution mechanism. One of the most difficult challenges is to design the scheduling algorithm of the push-server. This algorithm needs to determine the "broadcast scheduling map", namely which web items to broadcast and when. In this paper we study the approach where every constant period of time each proxy cache analyzes the requests it has received in the past and determines which web item it prefers to receive by broadcast and when. We formalize a related problem, called the "Cache Pre-filing Push" (CPFP) problem, analyze its computational complexity, and describe efficient algorithms to solve it.**

I. Introduction

Web caching is an important way to address the main problems of the WWW: Internet congestion delays, transmission cost and availability of web servers. A web proxy cache sits between Web servers and clients and stores frequently accessed web objects. The cache receives requests from the clients and uses the stored objects when possible in order to serve these requests. The traditional approach for a proxy to fill its cache when the content for a received request is not locally available was to access the original server, or another cache, using some inter-cache protocol like ICP [13], [14]. In such a case the proxy delivers one copy of the requested object to the requesting user (or proxy) and stores another copy locally for future requests.

In order to increase the availability of popular objects at proxy cache servers, and to reduce the bandwidth consumed by such servers, a new concept called Content Distribution Network (CDN) has been emerging. Loosely speaking, a CDN is architecture of network proxy servers, arranged for efficient delivery of web items. There are several approaches for implementing a CDN. These approaches differ mainly in their business model: who pay for the CDN and what for, and consequently in their implementation details: where the proxy servers are deployed, how are user's requests routed to these proxy servers, how do these servers get fresh web items, etc. One business model for a CDN is to increase the availability of web items distributed by certain content providers. In this model the CDN is a service provided to a group of subscribed content providers. Another business model is to reduce the costly bandwidth consumed by ISPs. An emerging CDN-based technology that serves this purpose is referred to as *cache pre-filling* [7].

With cache pre-filling, web items are pushed to the proxy cache servers, usually through a broadcast-based or a multicast-based distribution mechanism. The idea is to insert pre-selected URLs to the caches of N proxies that belong to a multicast group of the CDN, while encountering a cost which is significantly lower than the cost of N independent unicasts.

A popular distribution mechanism for a cache pre-filling is a satellite link, because such a link imposes a broadcast's cost of 1 unit, regardless how large N is. Figure 1 shows a typical satellite-based cache pre-filling system. The system consists of an intelligent "push-server", that collects web items from the Internet servers and broadcasts these items to the various proxy caches through a satellite link. The requests sent by the end users are routed to the nearby proxy cache. If the requested object is found, it is immediately sent back to the requesting users. If it is not found, the cache proxy forwards the request to the original server and receives the item from this server, either in unicast through a terrestrial link or in broadcast through the satellite link.

One of the most difficult challenges in this architecture is the scheduling algorithm employed by the push-server. This algorithm needs to determine the "broadcast scheduling map", namely which web items to broadcast, and when. One approach is to have the push-server sit on the data path between the proxy caches and the Internet. The push-server analyzes all the requests received from the various proxy servers and determines a broadcast scheduling map accordingly. The main advantage of this approach is that the proxy caches do not actively participate in the scheduling algorithm. Rather, they perform a standard proxy cache task of forwarding to the Internet all the requests that cannot be locally addressed. However, it does require that all the data paths from the different proxy caches will go through the push-server.

When this is not possible, the proxy caches must be involved in the process. In this approach, every constant

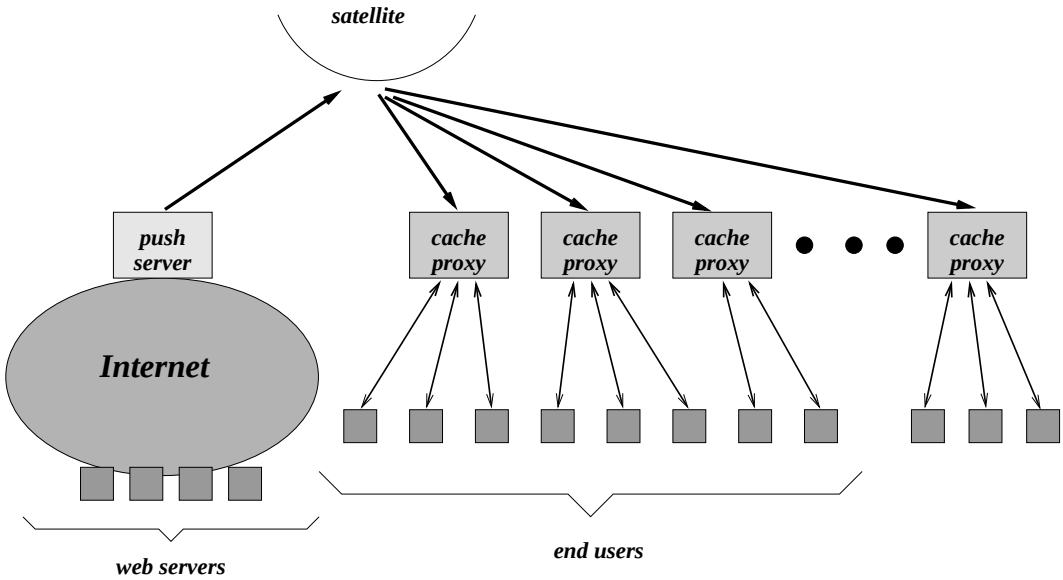


Fig. 1. A Satellite-based Cache Pre-filling CDN

period of time λ (e.g. $\lambda = 60$ seconds), each proxy cache analyzes the requests it has received in the last δ time period (e.g. δ=1 hour), and determines which web item it prefers to receive by broadcast and when. For instance, if a proxy cache recognizes a steady demand for a certain web item, it will request the push-server to broadcast this web item immediately when the copy it already has expires. The push-server then needs to compile all the requests received from the N proxies into a single broadcast scheduling map. The problem in this case is how to create the most profitable scheduling in light of possible collisions in the requests made by different proxies.

In this paper we propose a scheme based on the latter approach. In the proposed scheme the time axis is divided into fixed intervals, where every interval consists of T fixed time slots. Each time slot allows the push-server to broadcast one web item or a portion thereof. Before a slot starts, each proxy cache l, denoted p_l, sends to the push-server a profit matrix M_l. Entry $[i, j]$ in this matrix indicates the local profit for receiving by broadcast web item i in slot number j of the next time interval. The profit takes values between 0 and 1. When the push-server receives the matrices from all the N proxy caches, it computes a single scheduling vector F for the next time interval. This vector indicates which web item will be transmitted in every time slot of the next time interval.

The rest of the paper is organized as follows. In Section II we discuss related work. In Section III we analyze the problem of computing the most profitable scheduling when all the web items are of equal size of one slot. We than present an optimal algorithm for this problem, based on the concept of graph matching. In Section IV we extend the problem to handle the more realistic case where different web items have different sizes, and they therefore require different transmission times. We show that the resulting scheduling problem is NP-Complete, and propose efficient algorithms to solve it. In Section V we present simulation results for the various algorithms discussed in Section III and Section IV. Finally, Section VI concludes the paper.

II. Related Work

The algorithms presented in this paper are proven to be optimal, or optimal up to a constant factor, for the problem introduced in Section I, namely cache pre-filling through a broadcast channel. Many papers studied the problem of "broadcast scheduling" in the past. However, to the best of our knowledge none of them was in the context of the problem we consider. In this section we discuss some of these papers, and outline the differences between the problems they address and the problem we do.

In [2], the authors consider a system where the broadcasting station (the "push server" in our terminology) does not always have all data items available for broadcast. They developed a set of mechanisms that coordinate the process of broadcast scheduling with the process of locating and retrieving the data items to be broadcasted. However, since we consider in this paper a pre-filling system, an inherent assumption in our model is that the cache proxies tell the push server in advance what is the list of the data items they wish to receive during the next time interval. Moreover, in order to allow the push server to collect the required information, the proxy servers can send this information to the push server even 1 or 2 minutes in advance. This is because this information is determined based on long-period statistic collection, which is not af-

fected by events that take place during the time this information is collected by the push server.

The model presented in [1] and [11] is also different from the one we consider. Two main differences are as follows. First, in [1] and [11] it is assumed that the requests from the receivers are received by the broadcasting station asynchronously, and they are processed according to the order they are received. In contrast, in our model it is assumed that the push server receives all the requests before the beginning of each time interval, and it processes these requests together. One important consequence of this aspect is that in contrast to [1], there is no need for preemption in order to optimize the benefit. The second difference is that in [1] and [11] the broadcast channel is the only means through which information can be delivered to the receiving parties. Therefore, the penalty for delaying the broadcast of the data requested by one of the receivers is proportional to the time elapsed since the request is received. In contrast, in our system each receiver can receive over its "private unicast channel" all the information the push server did not broadcast. Therefore, the penalty for not broadcasting some information is proportional to the number of receivers that request this information through their private unicast channel. Moreover, whereas in [1] and [11] the broadcasting server must remember unsatisfied requests in order to accommodate them in the future, in our model such requests are ignored since they broadcasting server receives fresh requests before each time interval.

In [12] the authors distinguishes between two models: a pull-based and a push-based. In the former model, the receiving parties inform the broadcasting node about their exact requirements. This is similar to the models discussed in [1] and [11]. In the latter model, the receiving parties cannot inform the broadcasting station about their exact needs. In both cases, the target is to minimize the average delay of the receiving parties. Ref. [12] formulates the push-based problem as a deterministic Markov Decision Problem (MDP) and the pull-based problem as a stochastic MDP. In earlier works, a stochastic MDP model was used for designing periodic schedules with near optimal performance for the push-based problem [3], and scheduling policies were for the pull-based version were studied [9].

III. Fixed-size Web items

We consider the scenario described in the previous section, where the push-server has to determine the schedule for the next time interval given the benefit matrices from all proxy caches. Let the number of proxy caches be N, and assume that all proxy caches refer to the same set of web items W, where $W = \{w_1, w_2, \ldots, w_{|W|}\}$. Each proxy p_l creates a benefit matrix M_l, whose element in the ith row and jth column $M_l[i,j]$ indicates the expected benefit for proxy l from receiving web item w_i at time j, where $1 \leq i \leq |W|$ and $1 \leq j \leq T$. Recall that T is the length of the time period for which requests are received from the various cache proxies and a schedule is determined by the push-server. The expected benefit is a number between 0 and 1, i.e. $0 \leq M_l[i,j] \leq 1$. The overall benefit matrix M is a matrix in which every entry is the normalized sum of the benefit values of all the proxy servers, i.e., $M[i,j] = \frac{\sum_{l=1}^{N} M_l[i,j]}{N}$. Given this matrix M, the server needs to decide which web item should be broadcasted at each time interval such that the total benefit is maximized.

One can consider a different scenario where a web item that is not delivered through the satellite broadcast is retrieved using some other method, like unicast over a terrestrial link. In this case the objective is to minimized the "external" cost of retrieving these web items. One can set the benefit of an item to be this "extra" cost, and thus for a given request maximizing the profit is the same as minimizing the "extra" cost.

We start with a simple variant of the problem, where we assume that the time axis is divided into fixed time slots, and a broadcast of every web item takes exactly one slot. To address the issue of web item freshness, we assume that a new version of each web item is considered by the system as a different item, and that each web item is valid for at least T time slots[1]. Throughout the paper we also assume that all broadcasted web items during each time interval are kept in all the caches. Therefore, there is no merit in broadcasting a web item more than once during the same time interval. This assumption simplifies the computation of the local benefit matrices, because when a proxy station p_l computes entry $M_l[i,j]$ of such a matrix, it can assume that object w_i was not received during time slots $1 \cdots j-1$.

Definition III.1: A schedule is a function F from time slots to $\{W\} \cup \{NIL\}$. $F(t)$ indicates the Web item that is broadcasted at time t. The value of $F(t)$ is NIL if no item is scheduled for broadcast at t.

We can now formally define the Cache Pre-filing Push (CPFP) problem:

Definition III.2: Given a list of N cache proxies $p_1, p_2, \ldots, p_N$, where each proxy p_l has a benefit matrix M_l over a common set of web item $\{W\}$ for a period of T time units, find a schedule F such that each web item is selected at most once, and $B(F) = \sum_{t=1}^{T} \sum_{l=1}^{N} M_l[F(t),t] = \sum_{t=1}^{T} M[F(t),t]$ is maximized.

From the above discussion one can see that the only relevant benefit functions are these in which at some time step t_0 the value goes above zero, and it is not increas-

[1]This is a realistic assumption because in a real system cache-able web item are valid for at least 2 minutes whereas a typical value for T is 30 seconds.

ing after time t_0. However, the solution we present finds the optimal schedule without any restriction on the benefit functions.

A natural candidate to serve as a scheduling algorithm is the following greedy approach, referred to as the "maximum local benefit algorithm". Before each time step t, this algorithm inspects all the web items whose transmission at t would yield some benefit, and broadcasts the most profitable web item that has not yet been broadcasted. However, it is easy to realize that this approach does not yield the optimal strategy. Actually, for some input instances this greedy approach may perform very badly since it does not take into account the rate in which the benefit function decreases, and thus can choose a slightly higher value just to find out that the benefit of another web item has dropped down to zero. As an example, consider the following benefit matrix:

$$M = \begin{pmatrix} \epsilon & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \epsilon & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \epsilon & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \epsilon & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & 0 & \epsilon & 1 \\ 0 & 0 & \cdots & \cdots & 0 & 0 & \epsilon \end{pmatrix}.$$

The optimal solution for this benefit matrix is to select in time slot $t = 2, 4, 6, 8, \ldots$ item $t-1$, and in time slot $t = 1, 3, 5, 7, \ldots$ no item. Such a schedule would achieve a total benefit of $T/2$. However, the "maximum local algorithm" described above would select item t in every time slot t, achieving a total benefit of $T \cdot \epsilon$ only. This implies that the worst-case performance ratio of this algorithm is $O(1/\epsilon)$ where ϵ can be as small as one chooses. Note that this argument can be generalized to show that *any* on-line algorithm has an $O(1/\epsilon)$ worst-case lower bound[2].

Another possible algorithm is the "maximum *global* benefit algorithm". Like the "maximum *local* benefit algorithm", this algorithm can also be classified as a greedy algorithm. However, this algorithm makes a greedy selection based on the entire benefit matrix, rather than on the information of a given time slot only. This algorithm scans the whole benefit matrix and chooses the web item with maximum benefit that has not been chosen yet, provided that this transmission does not collide with the transmissions of previously-selected web items. This process is repeated until no more benefit can be achieved. This algorithm may have a performance ratio of 2. For instance, for the following benefit matrix

$$M = \begin{pmatrix} 1-\epsilon & 1 \\ 0 & 1-\epsilon \end{pmatrix},$$

the optimal solution yields a benefit of $2(1-\epsilon)$ whereas the "maximum *global* benefit algorithm" yields a benefit of only 1. To prove that 2 is the *worst-case* performance ratio of this algorithm, note that each item w_i selected by this algorithm to be broadcasted at t prevents from this algorithm the gain of at most two other transmissions: a transmission of w_i at another time slot t' and the transmission of another web item w_j at t. However, $M[i,t] \geq M[i,t']$ because otherwise the "maximum *global* benefit algorithm" would choose $M[i,t']$ rather than $M[i,t]$, and for the same consideration $M[i,t] \geq M[j,t]$.

However, the off-line problem can still be solved in polynomial time using the well known maximum-matching algorithm in bipartite graphs [8]. A matching in a graph $G(V,E)$ is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$ at most one edge of M is incident on v. M is maximum-matching in a weighted graph if for any other matching M', the sum of the weights of M is not smaller than the sum of the weights of M'.

We first construct a bipartite graph G_b as follows. The nodes of one set are the web items and the nodes of the second set are the time intervals. The set of edges is constructed such that each node that represents an item w_i is connected to each node that represents a time interval t. The weight associated with such an edge is $M[i,t]$. We can now prove the following Claim.

Claim III.3: There is a one to one correspondence between the matchings in G_b and the scheduling F where the cost of a matching is exactly the benefit of the correspondence schedule.

Proof: A feasible schedule in the given CPFP is a matching in G_b since at each time unit we broadcast at most one web item, and no web item is broadcasted more than once. On the other hand, a matching in G_b is an assignment of web items to time slot that satisfy the definition of a schedule. Since we assign a weight $M[i,j]$ to the edge (i,j) in G_b, the cost of the matching is exactly equal to the benefit of the schedule. ■

Using Claim III.3 we can now define the following algorithm for CPFP.

1. Construct the graph G_b;
2. Find and return a maximum-matching for G_b.

The complexity of the algorithm is essentially the complexity of perfect matching, namely $O((T+|W|)*T*|W|*\log(T+|W|))$. See [6] for a summary of the best known algorithms for this problem.

IV. Variable-size Web items

We now drop the assumption that all web items are of equal size. We associate an integer T_i with each web item w_i, that indicates the number of slots required for broadcasting w_i. In the profit matrix M, entry $M[i,j]$ indicates the profit gained from *starting* the transmission of item w_i at time j. Since the transmission of w_i after time

[2]With respect to the considered problem, CPFP, an on-line algorithm is an algorithm that knows at t only the t'th row of M.

$T - T_i + 1$ cannot be completed before the target time T, $M[i,t] = 0$ for every $t > T - T_i + 1$. The goal of the scheduler is to find a schedule that maximizes the gained profit.

Definition IV.1: A schedule F is a transmission vector that indicates which item starts being transmitted in which slot. If $F(i) = j$, then at time slot i the transmission of item w_j starts. A feasible schedule is a transmission vector that fulfills the following conditions:

(a) At each time slot at most one web item is transmitted.

(b) Each web item is transmitted at most once.

(c) Preemption is not allowed. This implies that the transmission of a web item must be completed before another item starts being transmitted. Namely, if $F[i] = j$ and $T_j > 1$, then $F[i+1, \cdots, i+T_j-1] =$ NIL.

The overall profit gained from a schedule F is denoted $B(F)$. As before, the problem is to find a schedule F such that $B(F) = \sum_{t=1}^{T} M[F(t), t]$ is maximized. However, unlike in the fixed-size case, the problem of an optimal schedule for the variable-size case is NP-Complete. The proof is obtained using a reduction from the Sequencing with Release Time and Deadline Problem (SRTD)[10]. Given an instance of the SRTD problem we construct an instance of the variable-size CPFP problem as follows. Let the set of jobs be W. For each job $i \in W$, $1 \leq i \leq n$, we construct a web item w_i with the following benefit function

$$B_i(t) = M[i,j] = \begin{cases} 1 & \text{if } r_i \leq t \leq d_i - p_i \\ 0 & \text{otherwise} \end{cases},$$

where r_i, d_i and p_i are the release time, the deadline, and the processing duration of job i. In addition, the broadcast time of web item w_i is equal to p_i. To finish the proof, it is sufficient to show that all jobs can be scheduled while satisfying the release time and deadline time constraints if and only if the optimal cache scheduler has benefit of $|W|$. Recall that $|W|$ is the number of jobs and thus the number of web items. If we have a web item scheduling with benefit of $|W|$, it implies that every web item w_i is broadcasted between time r_i and time $d_i - p_i$, and therefore meets its scheduling constraints. On the other hand, if there is a schedule that satisfies the constraints for every job i, the web items can be broadcasted according to this schedule, thereby achieving a benefit of $|W|$.

The "maximum *local* benefit algorithm" in this case is very similar to the one described in Section III for the fixed-size case. At time t, this algorithm scans the benefit matrix M and locates all the web items that have not been transmitted in the past whose transmission *starting at* t would yield some benefit. The benefit of each item is *normalized* by the length of the web item. The item with the maximum normalized benefit is selected for transmission. Assuming that the length of the selected item is δ, the algorithm continues by selecting a new transmission for time $t+\delta$. The worst-case performance ratio of this algorithm is $O(1/\epsilon)$, because the analysis in Section III for the fixed-size case is applicable also for the variable-size case.

The "maximum *global* benefit algorithm" for the variable-size case works as follows. It scans the whole benefit matrix and chooses the web item with maximum *normalized* benefit that has not been chosen yet, provided that this transmission does not collide with the transmissions of previously-selected web items. This process is repeated until no more benefit can be achieved. However, the worst-case performance ratio of 2 of this algorithm for the fixed-size case does *not* prevail for the variable case. Consider for example two items: item i whose length is 1 and profit is also 1, and item j whose length is T and profit is $T - \epsilon$. Since the normalized profit of i is larger, it will be selected by the "maximum *global* benefit algorithm", thereby achieving a total benefit of 1 rather than $T - \epsilon$.

Next we present a 2-approximation algorithm to solve the problem. The algorithm is an implementation of the scheduling algorithm proposed in [4], which is based on the "local ratio technique" presented in [5]. We begin with two definitions.

Definition IV.2: A transmission instance $I = (w_i, j)$ indicates that a web item w_i starts being transmitted at slot j. The merit of $I = (w_i, j)$, namely $M[i,j]$, is denoted $M(I)$. A transmission instance $I = (w_i, j)$ is said to end at slot $j + T_i - 1$.

Definition IV.3: Two transmission instances I and I' are said to have a conflict if they cannot appear together in a feasible schedule.

We can now present the algorithm. Notice that the merit matrix M is indeed a matrix since I is indicated by two indices, so the value of $M(w_i, j)$ is $M[i,j]$. The approximation algorithm works as follows:

1. Let the original merit matrix be M. Set $M_1 \leftarrow M$ and $i \leftarrow 1$.

2. Find in M_i the first transmission instance to end, and add it as I_i to a tentative schedule F. If two or more transmission instances meet the requirement select one of them arbitrarily.

3. Decompose M_i into two new merit matrices $M_{i+1}^{(1)}$ and $M_{i+1}^{(2)}$ in the following way:

(a) Copy from M_i to $M_{i+1}^{(1)}$ the transmission instance I_i and all the transmission instances in M_i that have a conflict with I_i. Set the rest of the matrix as 0, namely:

$$M_{i+1}^{(1)}(I) = \begin{cases} M(I_i) & I = I_i \text{ or } I \text{ has a conflict with } I_i, \\ 0 & \text{otherwise.} \end{cases}$$

(b) Copy from M_i to $M_{i+1}^{(2)}$, but decrease the merit of all the transmission instances that appear in $M_{i+1}^{(1)}$ (namely,

the transmission instance I_i and all the transmission instances in M_i that have a conflict with I_i) by the merit of transmission instance I_i ($M_i(I_i)$).

4. Generate a new merit matrix M_{i+1}. Set $M_{i+1} \leftarrow M_{i+1}^{(2)}$ then remove from M_{i+1} all the transmission instances whose merit is ≤ 0.

5. If M_{i+1} is not empty, set $i \leftarrow i+1$ and go to step 2, otherwise let $K = i$ and continue as follows.

6. Let the schedule created during this process be $S = [I_1, I_2, \ldots, I_K]$. We now generate from S a series of feasible schedules $[S_0, S_1, \ldots, S_K]$ in the following way:

(a) Let $S_K \leftarrow \phi$ and $i \leftarrow K-1$

(b) If $S_{i+1} \cup \{I_{i+1}\}$ is feasible, then $S_i \leftarrow S_{i+1} \cup \{I_{i+1}\}$.

(c) If $i \geq 1$, set $i \leftarrow i-1$ and go to step 6(b)

7. Return S_0 as a feasible 2-approximation schedule.

Note: matrices $M_i^{(1)}$ are not used by the algorithm. We have added them in order to facilitate the correctness proof later. The Algorithm needs to maintain only one matrix M that will play the roll of M_i for every i. The same argument holds for $S_0, S_1, \ldots, S_K$ as well.

Figure 2 depicts one iteration of the algorithm. Without lose of generality, let us assume that this is the first iteration. Hence, Figure 2(a) shows the original merit M_1. There are two transmission instances in this matrix that end before the rest of the instances: instance (a) whose merit is 10, and instance (i) whose merit is 15. The algorithm should choose one of them arbitrarily as I_1, and transmission instance (a) is selected. Now $M_2^{(1)}$ and $M_2^{(2)}$ are created (Figure 2(b) and Figure 2(c) respectively). Matrix $M_2^{(1)}$ contains all the transmission instances that appear in M_1. However, the merit of the transmission instances that have a conflict with instance (a) is set to 10 and transmission instances that do not have a conflict with transmission instance (a) is set to 0. Matrix $M_2^{(2)}$ contains all transmission instances that appear in M_1. However, the merit of the instances that have a conflict with transmission instance (a) is decreased by the merit of transmission instance (a) (10). Finally, Figure 2(d) shows matrix M_2. It contains all the instances from $M_2^{(2)}$, except those that have a non-positive merit. The instance selected by the algorithm as I_2 is (i).

The algorithm always stops since transmission instance I_i does never appear in M_{i+1}. In a trivial implementation the best bound on the running time would be $O((|W|T)^2)$ since for each instance we may be required to update the value of all other instances. This is still polynomial in the input size since we have an explicit benefit value for each web item per each time in the matrix M which is part of the input. The schedule S_0 returned by the algorithm is feasible since we start from a feasible schedule $S_K \leftarrow \phi$ and step 6(b) adds a transmission instance I_i to S_{i-1} if $S_i \cup \{I_i\}$ is feasible.

It remains to prove that S_0 is a 2-approximation. The proof can be found in [4]. However, due to the importance of the proof to the understanding of the algorithm, and since the proof in [4] is somewhat complex, we adopted it to our specific problem and present it in what follows. The proof is based on the Local Ratio Theorem:

Theorem IV.4 (Local Ratio) Let F be a set of constraints and let P, P_1 and P_2 be profit functions such that $P = P_1 + P_2$. Let X be an r-approximate solution with respect to P_1 and P_2, where $r \geq 1$. Then, X is an r-approximate solution with respect to P.

The proof for this Theorem can be found in [4], [5].

Claim IV.5: For every i, $1 \leq i \leq K$, $M_i = M_{i+1}^{(1)} + M_{i+1}^{(2)}$, holds.

Proof: There are three types of transmission instances that should be addressed:

1. $I = I_i$: since $M_{i+1}^{(1)}(I_i) = M_i(I_i)$ (by step 3(a)) and $M_{i+1}^{(2)} = M_i(I_i) - M_i(I_i) = 0$ (by step 3(b)), $M_{i+1}^{(1)} + M_{i+1}^{(2)} = M_i(I_i)$ holds.
2. $I \neq I_i$ and I has a conflict with I_i: since $M_{i+1}^{(1)}(I_i) = M_i(I_i)$ (by step 3(a)) and $M_{i+1}^{(2)} = M_i(I) - M_i(I_i)$ (by step 3(b)), $M_{i+1}^{(1)} + M_{i+1}^{(2)} = M_i(I)$ holds.
3. $I \neq I_i$ and I has no conflict with I_i: since $M_{i+1}^{(1)}(I_i) = 0$ (by step 3(a)) and $M_{i+1}^{(2)} = M_i(I)$ (by step 3(b)), $M_{i+1}^{(1)} + M_{i+1}^{(2)} = M_i(I)$ holds.

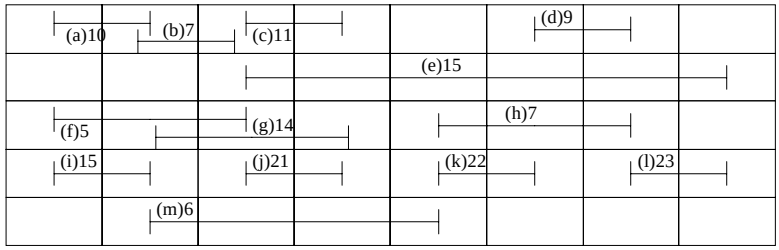
■

Definition IV.6 (I-maximal) A schedule S is called I-maximal if the following holds

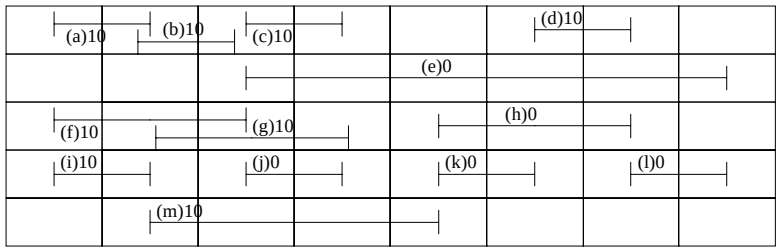
1. S is feasible
2. $(I \in S)$ *or* $((I \notin S) \wedge (S \cup \{I\}$ is not feasible$))$.

Claim IV.7: For every i, $1 \leq i \leq K$, a schedule S which is an I_i-maximal solution is a 2-approximation with respect to $M_{i+1}^{(1)}$.

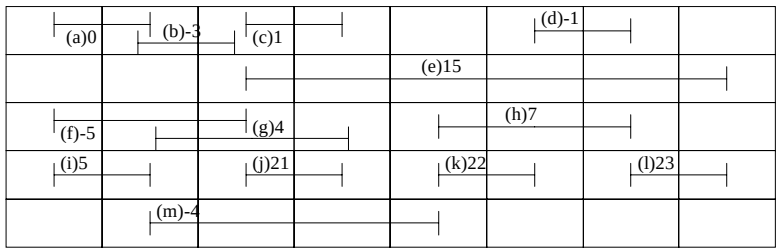
Proof: Let $A(I_i)$ be the set of all transmission instances in M_i associated with the same web item that transmission instance I_i is associated with. Let $B(I_i)$ be the set of all transmission instances in M_i that have a conflict with I_i and do not belong to $A(I_i)$. Note that $A(I_i) \cup B(I_i)$ is the set of all instances that have a conflict with I_i. Since $M_i(I_i) = M_{i+1}^{(1)}(I_i)$ by step 3(a), $M_{i+1}^{(2)}(I_i) = 0$ by step 3(b) . By the definition of $M_{i+1}^{(1)}$, only transmission instances $I \in A(I_i) \cup B(I_i)$ can contribute to the profit of a schedule S for $M_{i+1}^{(1)}$. Recall that all these instances have the same merit $M_i(I_i)$. To complete the proof it is sufficient to show that (a) since S is I_i-maximal solution it contains at least one transmission instance from $A(I_i) \cup B(I_i)$ so it has a merit of at least I_i; and that (b) no feasible schedule S' may contain more that one transmission instance from $A(I_i)$ and one transmission instance from $B(I_i)$, which implies that the maximum profit from a feasible schedule for $M_{i+1}^{(1)}$ is at most $2M_i(I_i)$. Part (a) follows directly from the fact that S is I_i-maximal. To prove part (b), note that no feasi-



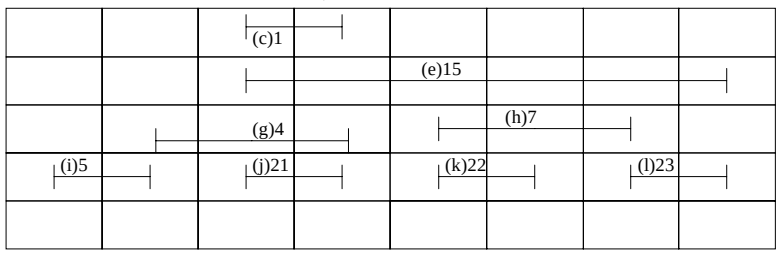
(a) Matrix M_i at the begining of step 2.



(b) Matrix $M_{i+1}^{(1)}$ at the begining of step 4.



(c) Matrix $M_{i+1}^{(2)}$ at the begining of step 4.



(d) Matrix M_{i+1} at the begining of step 5.

Fig. 2. Example of one iteration of the algorithm

ble schedule may contain two transmission instances of the same web item and there for S' may contain at most one item from $A(I_i)$. Now suppose that S' contains two items from $B(I_i)$: I' and I'' This implies that these two transmission instances do not intersect with each other. However, since they both belong to $B(I_i)$, both of them intersect with I_i. Therefore, at least one of them ends before I_i (see Figure 3), contradictory to the selection of I_i by step 2 of the algorithm. ■

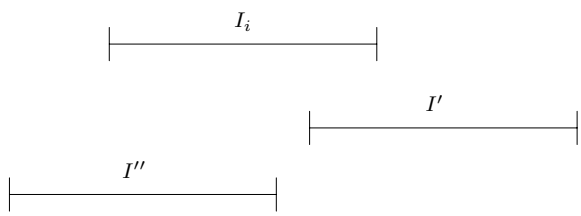


Fig. 3. The proof that S' contains at most one transmission instance from $B(I_i)$

Theorem IV.8: The return schedule S_0 is a 2-approximation with respect to M.

Proof: Recall that $M = M_1$. We prove the theorem by showing that for every i, $1 \leq i \leq K$, S_i is a 2-approximation with respect to M_{i+1}. The proof is by induction on i. We start with $i = K$. M_{K+1} is empty and $S_K = \phi$. Therefore, S_K is an optimal solution and

obviously a 2-approximation with respect to M_{K+1}. The induction hypothesis is that for $i = t$ it holds that S_t is a 2-approximation with respect to M_{t+1}. We now prove that S_{t-1} is a 2-approximation with respect to M_t. By the induction hypothesis S_t is a 2-approximation with respect to $M^{(2)}_{t+1}$ because M_{t+1} is generated by removing from $M^{(2)}_{t+1}$ non-positive values, and such values cannot increase the global merit. S_{t-1} is equal either to S_t or $S_t \cup \{I_t\}$ and by step 3(a) $M^{(2)}_{t+1}(I_t) = 0$. Thus, S_{t-1} is a 2-approximation with respect to $M^{(2)}_{t+1}$. By step 6(b) S_{t-1} is an I_t-maximal solution. Thus, by claim IV.7 S_{t-1} is a 2-approximation with respect to $M^{(1)}_{t+1}$. Since $M_t = M^{(1)}_{t+1} + M^{(2)}_{t+1}$ (claim IV.5), to conclude using Local Ratio (theorem IV.4) S_{t-1} is a 2-approximation with respect to M_t. ■

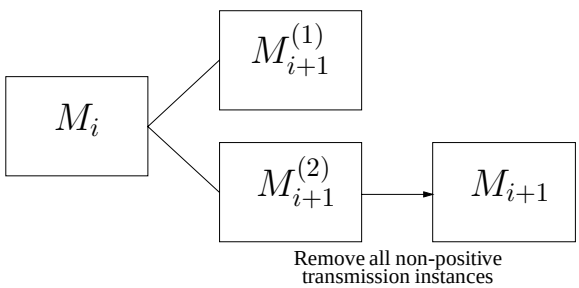


Fig. 4. Illustration of steps 3 and 4 of the algorithm

V. Simulation Results

In this section we present simulation results for the various algorithms presented in Section III and in Section IV for the fixed-size and variable-size web items respectively. Since for some of the discussed algorithms the theoretical analysis provides only worst-case guarantees, it is important to study the actual performance under realistic scenarios. In our simulation model, one can choose the following parameters:

- T - the size, in time units, of an interval.
- $|W|$ - the number of web items.
- N - the number of proxy caches.
- M - the benefit matrix.

As explained before, $M_l[i,t]$ represents the gain of proxy cache if item i is broadcasted at time t. Recall our assumption that there is a gain only from the first transmission of an item. The matrix M is generated in the following way. For each web item w_i and for every cache proxy p_l, we decide randomly if l is interested in item i or not. If it is interested, we choose randomly the first time $t \in [1..T]$ that the item is needed. In a real-system, t is the time when the current copy of the requested object expires at p_l. For this time t, we set $M_l[i,t] \leftarrow m$, where m is a random merit. For every time t', $t \leq t' \leq T$, we set $M_l[i,t'] \leftarrow m(1-p)^{(t'-t)}$. This implies that the merit decreases exponentially. The rationale behind this function is that in a real system if p_l does not receive the requested object by broadcast, it requests it by unicast as soon as the object is requested by some end host. Therefore, p indicates here the probability that the considered object is requested by an end host during a given time slot.

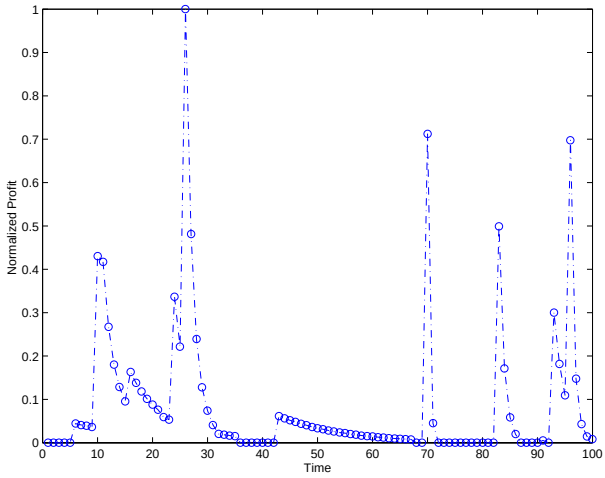


Fig. 5. An example for a profit function

As explained before, in order to get the global benefit matrix M we sum up the benefit matrices of all caches. An example of the resulting benefit matrix M for $N = 30$ cache proxies is depicted in Figure 5. One can see the effect of the different "peaks", for different proxies, and the sharp decrease in the benefit after this point. Clearly, if the number of cache proxy increases for a given T, the benefit function levels up, and for very large Ns it becomes much "smoother".

We start with the simulation results for the fixed-size case. We simulated the Following three algorithms:

- The "maximum-matching algorithm", as presented in Section III. Recall that this algorithm yields the optimal solution.
- The "maximum *local* benefit algorithm", as described in Section III. We have seen that this algorithm has an unbounded worst-case performance. However, its main advantage compared to the "maximum matching algorithm" is that it does not require the push-server to know the whole benefit matrix M in advance, but only the column for the next time slot t. This gives the proxy servers the freedom to submit their requests almost in real-time.
- The "maximum *global* benefit algorithm", as described in Section III. Like the "maximum matching algorithm", and in contrast to the "maximum local benefit algorithm", the "maximum global benefit algorithm" is an off-line algorithm. Recall that is has a worst-case performance ratio of 2.

Figure 6 depicts the profit vs. the number N of proxies. We consider two cases here: the case where the ra-

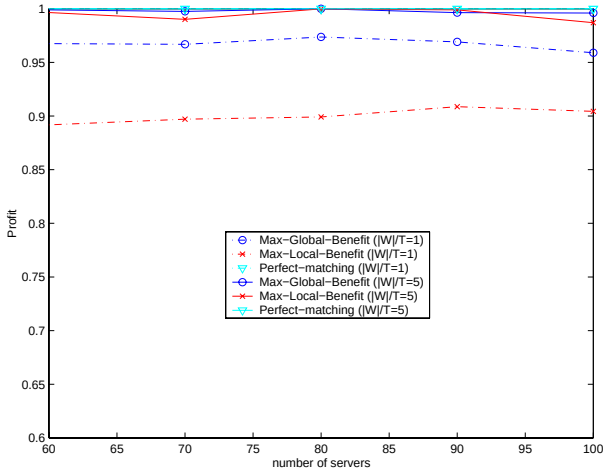


Fig. 6. The performance of the various algorithms for the fixed-size case as a function of the number of proxy stations

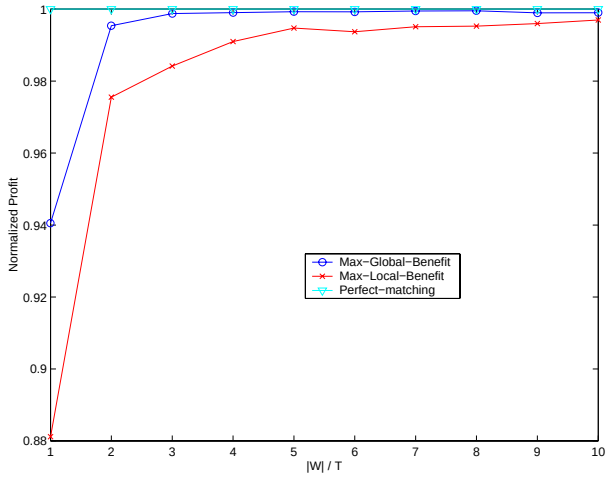


Fig. 7. The performance of the various algorithms for the fixed-size case as a function of $|W|/T$

tio between the length of the time period for which requests are received (T) and the size of the web item set ($|W|$) is 1:1, and the case where this ratio is 1:5. Since the maximum-matching algorithm yields the optimal solution, we view the profit of this algorithm as "1", and normalize the profits gained by the other algorithms accordingly. It is evident from the graph that when the ratio of $|W|/T$ increases, namely the size of the web item set increases, both the "maximum local benefit algorithm" and the "maximum global benefit algorithm" perform better. The explanation for this is as follows. When $|W|$ is large compared to T, the probability for these algorithms to loose significant profit due to a wrong selection is low, because there are always a lot of good choices. However, as $|W| \rightarrow T$, the effect of a wrong selection becomes bigger because the number of good choices is smaller.

As expected, the performance of the "maximum *global* benefit algorithm" is better than the performance of the "maximum *local* benefit algorithm". In fact, the "maximum global benefit algorithm" performs almost as good as the "maximum-matching algorithm". However, since this algorithm requires the same information used by the "maximum-matching algorithm", namely knowing the whole merit matrix M in advance, the only motivation to prefer it over the "maximum-matching algorithm" may be its simplicity. In contrast, the "maximum local benefit algorithm" does not require the push-server to know the whole M matrix in advance, and is also working in linear time (in the size of M). It therefore might be very useful in many systems. Another interesting observation from Figure 6 is that the number of proxies has no effect on the performance of the various algorithms.

We have seen in Figure 6 that the results of the two greedy algorithms are strongly affected by the ratio $|W|/T$. In order to study this aspect better, we tested the relative performance of the algorithms for more values of $|W|/T$. Figure 7 shows the normalized profit as a function of $|W|/T$. It is evident that when $|W|/T > 4$, this effect disappears, which indicates that at this point there are "enough" good candidates to broadcast even by a non-optimal algorithm.

Next we study the variable-size case. We considered again three algorithms:

- The "2-approximation algorithm", as described in Section IV.
- The "maximum *local* benefit algorithm" for the variable-size case, as described in Section IV.
- The "maximum *global* benefit algorithm" for the variable-size case, as described in Section IV.

Figure 8 and Figure 9 show the simulation results for this case. Since no algorithm achieves the optimal performance, the results are normalized according to those achieved by the "2-approximation algorithm". Figure 8 shows the normalized profit vs. $|W|/T$. In this graph, the length of each web item is randomly selected between 1 and 10 slots. It turns out that despite of their inferior theoretical worst-case performance, both the the "maximum *local* benefit algorithm" and the "maximum *global* benefit algorithm" outperform the "2-approximation algorithm" for small values of $|W|/T$, whereas for larger sets of web items the "2-approximation algorithm" performs better. The differences between the performance of the "maximum local benefit algorithm" and the performance of the "maximum global benefit algorithm" are bigger (up to 20%) for small values of $|W|/T$ and almost disappear for relatively large values of $|W|/T$.

Figure 9 shows the results of the three algorithms as a function of the maximum item size, for $|W|/T = 1$. We have seen in Figure 8 that when the maximum item size is 10, both the "maximum local benefit algorithm" and

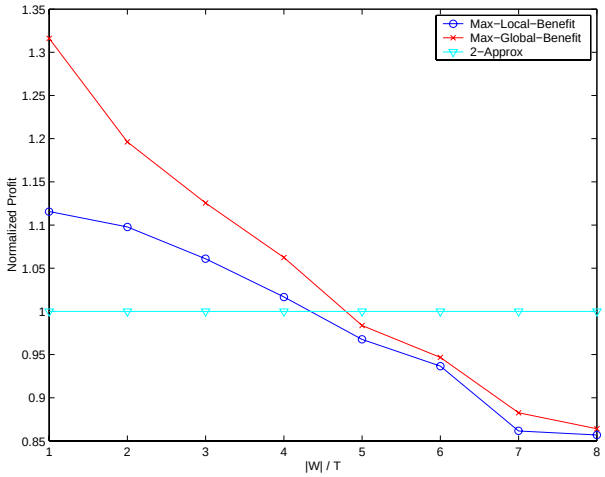


Fig. 8. The performance of the various algorithms for the variable-size case as a function of $|W|/T$

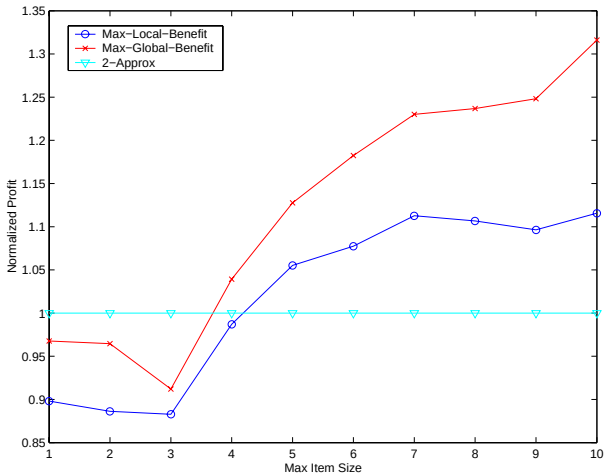


Fig. 9. The performance of the various algorithms for the variable-size case as a function of the maximum item size

the "maximum global benefit algorithm" perform better than "2-approximation algorithm". However, as the maximum item size decreases, the relative performance of the "2-approximation algorithm" improves. This is explained by the fact that as the maximum item size decreases, the number of intervals belonging to $M_I^{(1)}$ decreases as well, and thus the algorithm performance is reduced.

VI. Conclusions

Cache pre-filling is a new concept where web items are sent by a push-server to multiple proxy cache servers, through a broadcast-based distribution mechanism. One of the most difficult challenges with cache pre-filling is to design optimal Scheduling algorithm for it. In this paper we have studied the approach where every constant period of time each proxy cache analyzes the requests it has received in the past and determines which web item it prefers to receive by broadcast and when. We formalized a related problem, called the "Cache Pre-filing Push" (CPFP) problem, and studied this problem for two cases: the case where all web items are of fixed-size, and the case where the web items are of variable size. For the fixed-size items case, the CPFP problem can be solved in polynomial time using the well known maximum-matching algorithm in bipartite graphs. For variable-size web items the problem is NP-complete, but we developed a polynomial time approximation algorithm whose worst-case performance ratio is 2.

We then used simulation study in order to test how various algorithms for the CPFP problem perform. For the fixed-size case, we found the performance of the "maximum local benefit algorithm", that does not require the push-server to know the whole M matrix in advance, to be very competitive to the optimal performance. Therefore, this algorithm is probably the best choice for practical applications.

For the variable-size case, our simulations show that when the ratio $|W|/T$ is not too big, both the "maximum *local* benefit algorithm" and the "maximum *global* benefit algorithm" perform better than the "2-approximation algorithm", and thus no worst than twice the optimal solution.

References

[1] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pages 43–54, Oct. 1998.

[2] D. Aksoy, M. Franklin, and S. Zdonik. Data staging for on-demand broadcast. In *Proceedings of the 27'th VLDB Conference*, 2001.

[3] M. Ammar and J. Wong. On the optimality of cyclic transmission in teletext systems. *IEEE Transactions on Communications*, 35(1), Jan. 1987.

[4] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and schedualing. In *32nd ACM Symposium on the Theory of Computing*, 2000.

[5] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[6] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11:138–148, 1999.

[7] I. Cooper, I. Melve, and G. Tomlinson. Internet web replication and caching taxonomy. RFC-3040, Jan. 2001.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[9] H. Dykeman, M. Ammar, and J. Wong. Scheduling algorithms for videotex system under broadcast delivery. In *Proceedings of ICC*, pages 1847–1851, 1986.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[11] S. Hameed and N. H. Vaidya. Log-time algorithms for scheduling single and multiple channel data broadcast. In *Mobile Computing and Networking*, pages 90–99, 1997.

[12] C.-J. Su and L. Tassiulas. Broadcast scheduling for information distribution. In *INFOCOM*, pages 109–117, 1997.

[13] D. Wessels and K. Claffy. Application of the Internet Cache Protocol (ICP). RFC-2187, Sept. 1997.

[14] D. Wessels and K. Claffy. Internet Cache Protocol (ICP). RFC-2186, Sept. 1997.